Remarks on Verification Test Suites

Timothy Trucano
Optimization and Uncertainty Estimation Department
William Oberkampf and Martin Pilch
Validation and Uncertainty Processes

Sandia National Laboratories
Albuquerque, NM 87185

Phone: 844-8812, FAX: 844-0918
Email: tgtruca@sandia.gov

ASC V&V Workshop
January 9-12, 2007
Los Alamos National Laboratory
SAND2007-0565C

Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company,
for the United States Department of Energy's National Nuclear Security Administration
under contract DE-AC04-94AL85000.

I want to emphasize three aspects of test problems for a verification test suite:

I. There should be no question about why a test problem is defined: The purpose of the problem should be indisputable.

II. It should be rigorously established that a test problem is necessary: The relevance of the test problem should be indisputable.

III. Acceptable/unacceptable performance for a test problem can then be established: Pass/Fail on a test problem should be indisputable.

All three aspects should be established in a community context for a “Bi-Lab Test Suite” or a “Tri-Lab Test Suite.”

The point:

Stop arguing about purpose, relevance, and acceptable performance of codes on test problems, and start making sharp conclusions about passing the test that is presented by a test problem.

- **Do we want codes assessed by test problems or not?**
- **If not, then what, if any, is the purpose of a community test suite?**

Assessment means:

- Assessment requires:
  - Clear, unambiguous specification of the purpose of the test problems.
  - Clear, unambiguous specification of the relevance of the test problems.
  - If you can’t assess Pass/Fail on a test problem, you don’t have a sharply understood purpose and relevance.
- Assessment must be quantitative and rigorous:
  - Rigorous specification of the test
  - Verification norms (for comparing calculation with test)
  - Quantifying error on given meshes is as important as assessing order of convergence
  - Quantification of error (norm of test minus calculation) for given calculation setups (specifically grids).

Example: Sedov (notional)

- Purpose is to assess computational hydrodynamics
- Relevance: energy conservation, spherical blast waves in multi-dimensional calculations, agreement with similarity solution in L^p norms.
  - [Similarity solution raises well-known ambiguities in setting up the problem “properly.” Such ambiguities are irrelevant for energy conservation and spherical blast wave assessments.]
  - Pass = 0.1% energy conservation threshold (you tell me)
  - Pass = 0.01% deviation from spherical blast wave
  - Pass = 1% L^p -norm threshold compared to similarity solution
- There isn’t THE Sedov problem – there are many different ones even with an unambiguous initial condition:
  - 1-D spherical versus 2-D whatever versus 3-D whatever
  - Single-material versus multiple materials
  - Lagrangian versus Eulerian versus ALE versus AMR versus …
  - Shouldn’t they ALL run correctly?

Straightforward questions:

- How many test problems are enough?
- For what purpose?
- How simple should test problems be?
- How complex should test problems be?
- How can you ask about simplicity or complexity of test problems without thinking hierarchically about test problems?
- How much do we have to understand about test problems and why?
- We have a Code Comparison effort. Why do we also then need “Bi-Lab” or “Tri-Lab” verification test suites?
- Do you really want Pass/Fail assessment of performance of codes on test suites?

Less straightforward question:

- Are “Oracles” useful? – That is:
  - Suppose you have a test suite (it could be one problem) that has little or nothing of what we ask for above, but it comes with a rigorous and powerful Pass/Fail criterion.
  - That is, “passing” the test suite means the software is “correct,” and “failing” the test suite means the software is wrong, and “pass/fail” is unambiguous, and this has all been proven with mathematical rigor.
  - Who would use such a test suite (or single problem) and why?
- Use of Formal Methods is an example of this kind of oracle.
  - It’s a single test in principle: run your code through the Formal Method Engine (test) and it either proves or disproves that the software is correct – but you need not understand a single intuitive thing about how the conclusion is drawn.

Consider:

- Certainly one reason to have a community test suite is that its use can measure and improve the reliability of a code.
  - However, reliability involves users, not just codes.
  - There is a tacit knowledge component in code reliability, both on the part of code developers and of users.
  - Therefore – verification test suites speak to users, not just code developers.
  - Therefore, tests that act as oracles (neither users nor code developers tacitly understand them) don't improve the perception of reliability.
  - The absence of perception of reliability is the absence of reliability, at least for stockpile codes.
- Keep in mind – three stakeholder groups are serviced by test suites: (1) code developers; (2) users; (3) decision makers (evidence – ASC "indicators")

Strong Sense Benchmarks (test problems):

- Bill Oberkampf and I defined Strong Sense Benchmarks in 2002 as follows:
  - Exact, standardized, frozen, and promulgated definition of the benchmark.
  - Exact, standardized, and promulgated statement of the purpose of the benchmark. This addresses its role and application in a comprehensive test plan for a code, for example.
  - Exact, standardized, frozen, and promulgated requirements for comparison of codes with the benchmark results.
  - Exact, standardized, frozen, and promulgated definition of acceptance criteria for comparison of codes with the benchmark results. The criteria can be phrased either in terms of success or in terms of failure.

  [See Progress in Aerospace Science, V.38, 209-272 (2002)]
- Bill has recently elaborated this idea: “Design of and Comparison With Verification and Validation Benchmarks,” for the International Workshop “The Benchmarking of CFD Codes for Application to Nuclear Reactor Safety,” SAND2006-5376C, to be published.